

UNITED STATES PATENT APPLICATION

OF

Paolo NARVAEZ

residing at:
946 Mesa Oak Court
Sunnyvale, CA 94086

and a citizen of:
Spain

FOR

**Method and Device for the Classification and
Redirection of Data Packets in a Heterogeneous Network**

(Atty. Dkt. 085317-0307825; Client Ref. RZMI-P205-U)

PREPARED BY:

PILLSBURY WINTHROP LLP

Customer Number: **27498**

2475 Hanover Street

Palo Alto, CA 94304-1114

Phone: (650) 233-4500

Fax: (650) 233-4545

Attn.: Ross L. Franks, Reg. No. 47,233

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of priority under 35 U.S.C. §119(e) from U.S. Provisional Application No. 60/443,159 to Paolo Narvaez, filed January 27, 2003 and entitled “Classification of Packets in a Heterogeneous Data Redirection Device,” which is fully incorporated by reference in its entirety and for all purposes.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] Generally, the present invention relates to the telecommunications and digital networking. More specifically, the present invention relates to the classification of packets in a heterogeneous data redirection networking device.

Description of the Related Art

[0003] In the realm of digital networking and telecommunications, data is often assembled and then transmitted and received in certain discrete units known as packets. All or a specified number of the packets originating from the same source device, connection or application can be grouped together in a “flow.” Though the term “packets” is used in this discussion, “packets” may also refer to other discrete data units such as frames and so on. Network devices (e.g. switches, routers, etc.) that intercept and forward such flows or packets are often configured with a plurality of ingress ports (i.e., into which input flows arrive at the device) and a plurality of egress ports (i.e., from which input flows are routed outside the device). In this regard, and for purposes of the present invention, ports may be physical, logical or a combination of physical and logical. Further, ports may be bi-directional in nature such that they may serve as both ingress ports and egress ports. When an input flow or single input packet is received by a network device it could have been destined for output over just a single egress port or more than one egress port. An input flow/packet with just a single destination egress port is referred to as unicast, while a flow/packet that is destined for multiple egress ports is referred to as multicast.

[0004] Certain network devices may be classified as heterogeneous in that they may accept data (e.g., via ingress ports) of many different types and forward such data (e.g., via egress ports) in many different formats or over different types of transmission mechanisms. Examples of such devices include translating gateways that unpack data in one format and repackage it in yet another format. For purposes of the present invention, a heterogeneous network is considered the general case for those non-heterogeneous networks (i.e., those networks that accept and forward only one type of data over one type of transmission mechanism); as such, these non-heterogeneous networks are meant to be within the scope of this invention. When a two-port non-heterogeneous device merely forwards data from its one ingress port to its one egress port, there is less of a need to classify the packets. However, in devices where there are multiple types of ports (e.g., ingress, egress, ingress/egress combination, varying data formats, varying physical connections, etc.) and, where the physical ports may be combined or section into one or many logical ports, there is a critical need for packet classification.

[0005] **Figure 1** illustrates a heterogeneous network environment that provides for different types of data formats and marries different transport mechanisms. As will be evident to those skilled in the art, many different combinations of such formats and transport mechanisms (as well as many others) can be combined to form such a network environment. A network ring 100 may, for example, include a high capacity network such as a SONET ring and usually provides service to more than one customer. Such customers may further distribute the service(s) they receive via the network ring 100 to one or more nodes behind their own internal network. **Figure 1** shows nodes 110, 120, 130, 140, 150, 160 and 170. Nodes 140 and 150 are access the network ring 100 via the same Customer Premises Equipment (CPE) 145. Similarly nodes 130 and 170 access the network via CPE 135. The remaining nodes directly access the network ring 100. CPE 145 and 135 may, for example, be gateways that apportion transport mechanisms such as Ethernet or PDH (e.g., T1 lines, T3 lines, etc.) over the network ring 100, making use of the bandwidth given thereby. As mentioned above, network ring 100 can be a carrier-class network that may have a very large bandwidth, for example, such as 2.5 Gigabits per second (Gb/s).

[0006] Therefore, network ring 100 is generally not like a typical Local Area Network (LAN) service or a typical point-to-point leased line service. Recent efforts, however,

have sought to provide both of these types of services in an environment such as that shown in **Figure 1**. The first, called “Private Line Service” or “Ethernet Private Wire Service,” for example, is an attempt to create a secure and dedicated leased line type of mechanism. For instance, assume a dedicated connection-oriented service was desired between node 130 and node 110, which belong to the same organization B, yet might be separated by a long physical distance. A Private Line Service could be used to provide the node 130 to node 110 services. Private Line Service would create a point-to-point, dedicated interconnect between node 110 and node 130 even though both are segregated over the network ring 100.

[0007] Yet another service, known, for example, as a Transparent LAN or Ethernet Private LAN Service, is an attempt to create a virtual Local Area Network (LAN) out of those nodes that belong to the same customer. For instance, if nodes 120, 140, 150 and 170 (i.e., all belonging to the same customer organization A) were part of a virtual LAN using the network ring 100, they would appear to one other as being on the same customer A LAN, even though node 120 might be separated a great distance from nodes 140, 150 and 170. Further, the nodes would appear on the same LAN even though all four nodes might be operating under different transport mechanisms (e.g., if Transparent LAN services were enabled). Likewise, another Transparent LAN service could enable nodes 110, 130 and 160, which all belonging to customer organization B, to appear to be on B’s LAN.

[0008] Yet another challenge of provisioning such services over network ring 100 is from the standpoint of the CPEs, such as CPE 135 and CPE 145. The CPE must consider provisioning the services so that the nodes (e.g., end client or users) can accurately be identified according to their network flows and be distinguished from one another. Since the actual network is not precisely connection-oriented, the CPEs should be capable of distinguishing data units belonging to organization A from those belonging to organization B. Further, where both Private Line and Transparent LAN services are present and both seek to be provisioned on the same CPE, such services should be distinguished by the CPE. When a CPE also has the task of allocating its resources, scheduling and routing data units, determining egress ports and so on, packet classification becomes much more difficult. Further, where a CPE might support the provisioning of T1 lines to one node and Gigabit Ethernet or 10/100 Ethernet to another node, the CPE must be able to classify and distinguish among such packet types internally.

[0009] While CPEs 135 and 145 could simply be built with many different physical ports and large memories, such a CPE is cost-inefficient to the customer. Further, such a CPE does not scale well where the customer desires many different channels or logical ports over the same physical ingress or egress port. Recently, there are efforts underway to provide scalable CPEs that can operate on less hardware and thus, with less cost and complexity than their predecessors, but while still providing better performance than their predecessors. However, in such efforts, classification of data becomes important as functions such as prioritizing and scheduling data units and allocating limited CPE physical resources (e.g., memory and processing cycles) come to the fore. Further, where logical and physical ports are bidirectional in nature, having both egress and ingress capability, scheduling and allocating become vital to the proper operation of the network.

[00010] Thus, what is needed is a networking device with a data classifier that can cheaply and efficiently be used for managing, queuing, processing, scheduling and routing data units between multiple ingress/egress ports and between varying transmission media and transmission formats.

SUMMARY OF THE INVENTION

[00011] What is disclosed is a system and method for classification of data units in a network device that acts to bridge heterogeneous networks, provides many different services and provisions many different transport mechanisms. The data classifier, which is one subject of various embodiments of this invention, generates an ID internally used by the network device in managing, queuing, processing, scheduling and routing to egress the data unit. This device-internal ID enables the device to accept any type of data units from any physical/logical ports or channels and output those data units on any physical/logical ports or channels that are available. It also enables the device to identify data units used in Private Line Services and Transparent LAN services, among others.

[00012] The data classifier in accordance with the various embodiments of the invention is configurable and will behave differently depending upon the physical port to which it is interfaced. Within a given port, further, the data classifier can distinguish among physical or

logical ports and can be further configured to behave differently, if needed, according to the characteristics of the physical or logical port.

[00013] In at least one embodiment of the invention, the classification consists of tag extraction and then tag lookup. The tag lookup generates a flow ID and a priority ID that are combined to give the device-internal queue ID. In devices that provide link aggregation, a balancing component can enable generation of a balanced flow ID as well.

BRIEF DESCRIPTION OF THE DRAWINGS

[00014] These and other aspects and features of the present invention will become apparent to those ordinarily skilled in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures, wherein:

[00015] **Figure 1** illustrates a heterogeneous network environment that provides for different types of data formats and marries different transport mechanisms;

[00016] **Figure 2** illustrates at least one embodiment of the data classifier as deployed in a network device according to the present invention;

[00017] **Figure 3** illustrates a flowchart of a classification technique according to at least one embodiment of the invention;

[00018] **Figure 4** illustrates a block diagram of a classifier according to at least one embodiment of the invention;

[00019] **Figure 5** illustrates a block diagram of a tag extraction unit according to at least one embodiment of the invention;

[00020] **Figure 6** illustrates a diagram of a tag lookup engine according to at least one embodiment of the invention; and

[00021] **Figure 7** illustrates an embodiment of the invention in which both private line and private LAN services can be accommodated in a single classifier device, and thus in the same network hardware.

DETAILED DESCRIPTION OF THE INVENTION

[00022] The present invention will now be described in detail with reference to the drawings, which are provided as illustrative examples of the invention so as to enable those skilled in the art to practice the invention. Notably, the figures and examples below are not meant to limit the scope of the present invention. Where certain elements of the present invention can be partially or fully implemented using known components, only those portions of such known components that are necessary for an understanding of the present invention will be described, and detailed descriptions of other portions of such known components will be omitted so as not to obscure the invention. Further, the present invention encompasses present and future known equivalents to the known components referred to herein by way of illustration. The attached Appendix forms a part of the present disclosure and is incorporated herein by reference.

[00023] Figure 2 illustrates at least one embodiment of the data classifier as deployed in a network device according to the present invention. A network device 200 is shown as having a plurality of physical ports 205, 206, 207, 209 that can interface to varied and/or different transport mechanisms. Each of the ports 205, 206, 207 and 209 may also support one or more logical ports or channels. In this embodiment, port(s) 205 interfaces to a typical Ethernet service, such as a 10/100 network; port(s) 206 interfaces to a higher capacity GigE, or Gigabit Ethernet service; port(s) 207 interfaces to a PDH type services, such as T1 or T3; and ports 209 can be SDH-capable ports that interface to a carrier-class service, such as SONET. This configuration is intended to show one of many possible supported network interfaces and configurations, and is only intended to exemplify a complex network interface device 200 that supports a wide variety of networks, transmission/transport schemes and associated protocols.

[00024] As shown in Figure 2, a classifier 210 according to the present invention generates a Queue ID 220, which is internal to the device 200. The Queue ID 220 is composed of an internal Flow ID and a Priority ID (discussed further, below). The Queue ID 220 is used by device 200 to decide on which queue to store any given ingress packet before it gets scheduled (also discussed further, below). The classifier 210 includes a set of tag extractors 218, a tag memory 214 and a tag lookup 216.

[00025] For example, the classifier 210 can be coupled to the ports available on the device 200 (e.g., ports 205, 206, 207 and 209) to accept a first portion of every packet that ingresses on these ports. Each of the tag extractors 218 interfaces can be associated with one of the ports 205, 206, 207 and 209 and can check the first portion of every packet to see if the right type of packet (i.e., one that is compatible with that port's configuration) is received. Tag extractors 218 (discussed further below) use this first portion of the packet (e.g., in some embodiments, the first 32 or 64 bytes) to build a set of tags. The generated tags include, among other things, information on the customer and external flow/service (i.e., Transparent LAN), Media Access Control (MAC) source and destination, and link aggregation conversation information. The generated tags can then be stored in tag memory 214.

[00026] Once the tags for a packet are completed, the tag lookup engine 216 will read the tags serially from the Tag Memory 214 and perform a lookup. The lookup returns both a Flow ID and Priority ID (and if link aggregation is used, a balanced Flow ID), which are then combined to make up the Queue ID 220. One embodiment of a tag lookup engine, such as engine 216, is discussed in detail below.

[00027] **Figure 3** illustrates a flowchart of a classification technique according to at least one embodiment of the invention. As shown in **Figure 3**, a packet that ingresses into one of the many ports of the device is received 310. The packet type of the received packet is checked 312. If the checked packet type does not match the port type for which the packet was configured or to which the packet is destined 312, then a tag error is generated 315 and processing ceases 317 since the received packet is errant. If the packet type of the port over which the packet was received matches the port type, then tags are extracted 320. If the tag extractor finds an error with a tag during extraction, default tag values can be assigned instead (as discussed in further detail, below).

[00028] After the tags are extracted 320, according to this embodiment, the tags can be written to a memory so that they can be accessed at a later time 330. A lookup using the tags is then performed, which first includes a flow tag lookup and a MAC destination address tag and MAC source address tag lookup 340. Based on these tags, an output Flow ID is reconciled 350. If there is link aggregation present 360, then a Balanced Flow ID is also generated 365. Next,

after obtaining the Flow ID and/or the Balanced Flow ID, a lookup for the Priority ID tag is performed 370. The Flow ID or, if generated, the Balanced Flow ID, is then combined with the Priority ID 380. The result of this combination, the Queue ID, is then output from the classifier 390. The process then repeats itself for subsequently received ingress packets. Details for performing the processes outlined in **Figure 3** are discussed in greater detail below.

[00029] **Figure 4** illustrates a block diagram of a classifier according to at least one embodiment of the invention. As shown in **Figure 4**, classifier 400 consists of a Tag Extraction Unit 420, a Tag Lookup 430 and a Configuration Table 410. The Tag Extraction Unit 420 consists of a plurality of instances of a tag extractor, one for each port in the device (see **Figure 5** and associated description). The Tag Lookup 430 consists of a sequence of lookup operations that are performed on the extracted tags generated by the Tag Extraction Unit 420. The Tag Extraction Unit 420 (and its associated extractors) operates differently on each of the available ports that interface to the classifier 400. This per-port configurability of the tag extraction mechanism is handled by a Configuration Table 410. Each of the ports has an entry in the Configuration Table 410 which specifies how a tag should be extracted (by the Tag Extraction Unit 420) for packets at that port. Configuration Table 410 would include information such as parity bit selection, start and end position of tags and fields as well as length of tags and fields when needed.

[00030] **Figure 5** illustrates a block diagram of a tag extraction unit according to at least one embodiment of the invention. As shown in **Figure 5**, the Tag Extraction Unit 500 can consist of a number of packet tag extractors 510, for example: a GE (GigE) packet extractor 512, a ME (Ethernet) packet extractor 514, a SDH (SONET) packet extractor 516, and a PDH packet extractor 518. However, fewer or additional packet extractors might be included. The Tag Extraction Unit 500 can also consist of a tag extraction configuration block 410, an extraction memory 214, and an extraction mask block 540.

[00031] In this exemplary embodiment, each of the tag extractors 510 can consist of numerous separate extraction components, for example management flag extraction, error flag extraction, type tag extraction, flow tag extraction, MAC tag extraction, priority tag extraction, and balancer tag extraction, each is discussed in further detail below.

Management flag extraction

[00032] The management flag can be used to distinguish a management packet from other packets and can be as simple as a single bit. Packets received from all types of ports might, for example, be accompanied with an Extract Management Indicator (EMI). When the EMI is asserted at the start of classification (SOC), and when the EMI is enabled by the extraction configuration for that logical port, the received packet can be labeled as a management packet.

Error flag extraction

[00033] The error flag can be used to indicate errant packets. For example, a 3-bit error flag can be used to mark that a received packet is incorrectly extracted, where, for example:

[00034] Bit 2 (flow error) of the error flag can be set if either the type tag or flow tag extraction configurations has an error, or if the end of classification (EOC) occurred before the packet type tag or flow tag extraction was complete;

[00035] Bit 1 (MAC error) of the error flag can be set if the MAC tag extraction configuration has an error, or if the EOC occurred before the packet MAC tag extraction was complete; and

[00036] Bit 0 (balancer error) of the error flag can be set if the balancer tag extraction configuration has an error, if the balancer tag is configured to be zero length, or if the EOC occurred before the packet balancer tag extraction was complete.

Type tag extraction

[00037] The type tag extraction can obtain information used to verify whether the arriving packets are of the same type as any given logical port. The flow, MAC, priority, and balancer tag extraction uses the same tag extraction configuration for every packet arriving on the same logical port. If a different type of packet arrives on a logical port, the packet will be incorrectly extracted and misclassified.

[00038] The extraction can be accomplished, for example, by obtaining bytes that contain, for example, 0 to 16 contiguous bits belonging to the first 64 bytes of a packet. The position of the first bit and the tag length is configurable per logical port (via the tag

configuration 410). The value of these extracted bits should match that of the pre-configured value for that port. If not, the packet type is probably not valid, and the packet can be marked as an error packet. A configuration error is flagged for the received packet if its logical port is configured to extract a type tag with length greater than 16 bits, or if the starting bit plus the length falls outside of the first 64 bytes. As should be obvious to those skilled in the art, this extraction can easily be applied to byte-lengths of packets and with varying bit-lengths of extractions. An extraction error can be flagged for the received packet if there is a type tag extraction configuration error for that logical port, or if the EOC occurred before the extraction was complete.

Flow tag extraction

[00039] The flow tag obtains information used to identify a customer flow. The extraction is accomplished, for example, by obtaining 2 non-overlapping fields of contiguous bits up to 64 total from the first 64 bytes of a packet. The position of the first bit and length of both fields are configurable per logical port (via the tag extraction configuration 410). The 10-bit Logical Port ID (LPID) from which the packet was received can be appended in front of the two fields to form the flow tag.

MAC tag extraction

[00040] The MAC tag obtains information used for MAC destination lookup and source learning. The extraction is accomplished, for example, by obtaining 12 contiguous bytes from the first 64 bytes of a packet. The position of the first byte is configurable per logical port. The first 6 bytes are the MAC destination address, and the last 6 bytes are the MAC source address. A configuration error can be flagged for the received packet if its logical port is configured to extract MAC tag with the starting byte plus 12 falls outside of the first 64 bytes. A MAC error can be flagged for the received packet if there is a MAC tag extraction configuration error for that logical port, or if the EOC occurred before the extraction was complete.

Priority tag extraction

[00041] The priority tag extraction obtains information used to determine the priority of the received packet. The extraction can be accomplished, for example, by obtaining 3

contiguous bits from the first 64 bytes of a packet. The position of the first bit is configurable per logical port. The priority tag will default to all zeros if the starting bit plus 3 falls outside of the first 64 bytes, or if the EOC occurred before the extraction was complete.

Balancer tag extraction

[00042] The balancer tag extraction obtains information used to determine the link aggregated customer flow. The extraction is accomplished, for example, by obtaining the 5-bit CRC (Cyclic Redundancy Check) of up to 15 contiguous bytes from the first 64 bytes of a packet. The position of the first byte and the length of the tag are configurable per logical port. The CRC function can be, for example, a polynomial function, $x^5 + x^2 + 1$, with an initialization value of all ones. While this exemplary polynomial is generally used for a USB token ring, other functions can also be employed. A configuration error can be flagged for the received packet if its logical port is configured to have the starting byte plus the length falls outside of the first 64 bytes. A balancer error can be flagged for the received packet if there is a balancer tag extraction configuration error for that logical port, or if the length is zero, or if the EOC occurred before the extraction was complete. The balancer tag will default to all ones, the initialization value, if a balancer error is flagged.

[00043] The tag extraction configuration block 410 can include all of the information necessary for determining where in a packet to obtain information for building a tag that is valid for the port over which the packet was received, and to which the packet might egress. Each port in the system can have a different packet stream and thus, tags for packets originating in that port should be constructed differently.

[00044] The memory 214 can, for example, store the resultant tags as the extractor builds them and store part of the extraction configurations at the EOC. When a tag lookup request is received, the memory 530 can retrieve both the extraction configuration from the tag extraction configuration 410 and the tag(s). The memory 530 can be logically organized into 369 logical FIFOs (if, for example, there were a total of 369 ports in the system). Each FIFO can hold a number of tags for each logical port (e.g., 4 tags per port). New tags created by the tag extractors 510 can be written to the tail of the FIFO corresponding to each tag's logical port. Tags to be looked up might be read out from the head of the FIFO.

[00045] The Tag Extraction Mask 540 can be, for example, a mechanism for stripping out unnecessary information from tags stored in the memory 530 when they are ready to be transferred from the tag extraction unit 500 (or, for instance, Tag Extraction Unit 420) to the Tag Look-up (for instance, Tag Look-up engine 430).

[00046] **Figure 6** illustrates a diagram of a tag lookup engine according to at least one embodiment of the invention. Once a tag is stored in the tag memory, the tag lookup engine 600 in the classifier can fetch the tag (e.g. using the appropriate tag extraction mask 540) and perform a search using one or more of the 5 fields stored in the tag (e.g., input flow ID, output flow ID, customer ID, MAC DA and MAC SA). When the full search is completed, the tag lookup engine 600 can output a Flow ID 674 of the packet, as well as a Priority ID 672. When link aggregation is used, the lookup engine can also output a Balanced Flow ID 676. The Balanced Flow ID can be used, for example, to perform load-balancing of a single Ethernet logical port over multiple physical ports. A Queue ID 678, which is the output of the classifier, is then generated by combining the Flow ID 674 (or Balanced Flow ID 676) and the Priority ID 672.

[00047] According to this exemplary embodiment, the flow tag lookup engine 610 can be implemented using a binary search tree. The values in the tree can be computed by software and downloaded into the device that integrates the classifier. These values can be changed dynamically by software or firmware. A shadow set of values can also be present and the switchover can happen hitlessly. The binary search tree can be implemented in any number of ways, many of which are well-known in the art.

[00048] As shown in **Figure 6**, the upper-left block of the tag lookup engine 600 is the flow tag lookup 610. The flow tag lookup 610 includes the task of finding the input flow ID, output flow ID and customer ID of a given packet (or Tag for that packet). It can accomplish this by using the previously extracted flow tag and finding the corresponding values. This correspondence can be based on an exact match or a set of ranges.

[00049] The output of the flow tag lookup 610 includes of a customer ID, an output flow ID and an input flow ID. The customer ID can be, for example, a value identifying which one of a group of customers (for instance, 256 customers) owns the packet (e.g., the customer ID could be the same as a carrier Virtual LAN tag). The output flow ID can indicate the internal

flow to be used for routing the packet internally to the correct egress port. The input flow ID is functionally equivalent to the output flow ID, but for transmission in the reverse direction (i.e., the internal flow/path to be used to route the return packet back to the input). The input flow ID is needed in order to do learning (discussed further, below).

[00050] In many cases, the flow tag lookup 610 may not be sufficient to accurately route a packet to the correct destination port. For example, the output flow ID from the flow tag lookup 610 might simply provide information to broadcast the packets to all remote hosts belonging to the same customer (e.g., multicasting). In such cases, a more precise destination lookup using MAC addresses is required.

[00051] The MAC destination address (DA) lookup 620 uses the customer ID (from the flow tag lookup 610) as well as the MAC DA Tag to find a more precise output flow ID. The output of the MAC DA lookup 620 will only be valid (i.e., valid over the output flow ID from the flow tag lookup 610) if the MAC address has been previously learned or programmed by the device (more on learning below). The MAC DA lookup 620 uses a hash table 625 with an appropriate number of entries (e.g., 256 entries, etc.) to store learned MAC addresses (along with their corresponding customer ID information). Hashing conflicts are resolved by using a list that is N entries deep. The output of the hash table 625 is the output flow ID used to internally route the packet to a proper, or better, output port.

[00052] If the MAC DA lookup 620 returns a valid result, it becomes the Flow ID of the packet. On the other hand, if it returns a null result (e.g., if the MAC address has not yet been learned or programmed), the default output flow ID from the flow tag lookup will be used instead. This logic is controlled by a selector mechanism 640, which uses a signal indicating whether the MAC address lookup was successful, or valid (i.e. if the MAC address had been previously learned or programmed, for example, the lookup would be successful, otherwise not).

[00053] MAC source address (SA) learning 630 can be used to associate a given MAC address and customer ID with a resulting output flow ID. Whenever a new packet arrives, the MAC source address tag of the packet can be used for learning. In this case, the source MAC address and the customer ID (e.g., provided by the flow tag lookup engine 610) will be associated with the input flow ID (e.g., also provided by the flow tag lookup engine 610). The

values can be written into one of the entries in the hash table 625. After a programmable time interval, the previously learned entries will age and will no longer be used by the MAC DA lookup engine 620. The aging time interval can be, for example, 16 clock cycles.

[00054] Unlike the learning schemes of the typical Private LAN of today, which performs on-the-fly learning on a per-port basis, the MAC SA learning 630 of the present invention is a per-flow learning scheme. According to an exemplary embodiment, each learned customer/MAC combination can be associated with a flow, instead of a port as is typical today. Thus, once the customer/MAC lookup is performed via the MAC SA learning 630, much more information is available, other than merely to which egress port(s) the packet is destined. For example, the input flow ID, output flow ID, MAC SA/DA, customer ID can all be available to the network, which equates to a more detailed granularity in the data handling. This detailed granularity allows the network device of the present invention to provide meaningful quality of service (QOS) management. As contrasted to the technologies of today, which do not allow for QOS management at such a network device.

[00055] Whenever link aggregation is used, a group of physical ports are programmed to act as a single logical port. In such cases, an additional balancer tag lookup 650 is utilized to distribute packets across the different physical interfaces that constitute that one logical interface. The balancer tag lookup 650 should make sure that any two packets belonging to a single conversation are sent on the same physical interface. This helps to ensure that all packets belonging to the same conversation will be delivered in order. On the other hand, the balancer tag lookup 650 should also load-balance the traffic from different conversations effectively across all applicable physical interfaces of that one logical interface to maximize bandwidth requirements.

[00056] The balancer tag lookup 650 uses a function to obtain a hash value of the conversation tag. The hash value is then used to index a table that indicates the physical interface (or interfaces) to be used by the packet. The new flow ID obtained by the balancer tag lookup 650 is called a Balanced Flow ID 676. Since two equal conversation tags produce the same hash value, all packets from the same conversation are guaranteed to have the same balanced flow ID 676 and hence use the same physical port(s).

[00057] The flow ID 674 (i.e., from the flow tag lookup 610 or MAC DA lookup 620) can be used as an input to a priority tag lookup 660 to find the priority ID 672. For example, a k-bit flow ID 674 can be combined with the n-bit priority tag to make a (k+n)-bit data item within the priority tag lookup 660. The priority tag lookup 660 can, for example, consist of a large table with $2^{*(k+n+1)}$ entries, where each entry can have, for example, a 2-bit value. The looked-up 2-bit value can then become the internal Priority ID 672 of the incoming packet.

[00058] For non-error and non-management packets, the flow ID 674/676 (output of the flow tag lookup 610 or MAC DA lookup 620 blocks, sometimes modified by the balancer tag lookup 650) and the Priority ID 672 (output of the priority tag lookup 660) are concatenated. For example, the bits from the Flow ID 674/676 are the most significant bits while the Priority ID 672 forms the least significant bits. This combined value can be the Packet Queue ID 678. It can determine into which of the external packet queues the packet of interest will be written. This is the classifier output.

[00059] If a given flow tag has been marked as an error, the output of the lookup will also be an error indication, regardless of the contents of the tag. If the tag is marked with an error, the output of the lookup engine will have the error bit enabled. This invalidates whatever Queue ID 678 value is generated and indicates the corresponding packet should be dropped.

[00060] The system and methods described above can simultaneously be applied to private line services and private LAN services, as required. In systems that support both of these services, keeping track of the flow internally within the device through which the packet is being forwarded can help ensure that the packet is placed on an egress port such that it maintains its membership to a particular private LAN or is routed along the path specified by the private line. **Figure 7** illustrates an embodiment of the invention in which both private line and private LAN services can be accommodated in a single classifier device, and thus in the same network hardware. This also has the advantage of accommodating different types of private LAN mechanisms, such as VLAN or MPLS (Multiple Protocol Labeling Service).

[00061] The classifier 700, which is similar to the classifier mentioned in other embodiments of the invention, can be broken down into two functional components. As a packet's tags are extracted in the classifier 700, it is unknown whether the packet belongs to a

private line service or belongs to a private LAN service. The first functional prong of the classifier 700 is a search function 710 (which is used, for instance, in the flow tag lookup 610 discussed above). The search function 710 utilizes a lookup capability to determine whether the packet belongs to a private line service or a private LAN service. The search function 710 returns a default route, which is the actual route the packet will take if the packet is a private line packet. If the search function 710 determines that the packet belongs to a private LAN service, then the customer ID and MAC Destination Address tag can be input to a hash table 725 (e.g., in a similar manner as 620, 625 and 630, discussed above), which generates a different flow ID for the private LAN service data. The flow ID from search 710 and the flow ID from hash table 725 are then input to a selection mechanism 740 (e.g., such as selector 640), which decides which of these flow IDs is the appropriate route and hence, the appropriate flow ID. The appropriate route may be the best or most efficient route and/or the route that satisfies private LAN characteristics and/or requirements. Such a scheme can be implemented in the hardware shown in **Figure 6**, for example, since both a search table and hash table can be utilized.

[00062] Although the present invention has been particularly described with reference to the preferred embodiments thereof, it should be readily apparent to those of ordinary skill in the art that changes and modifications in the form and details thereof may be made without departing from the spirit and scope of the invention. For example, those skilled in the art will understand that variations can be made in the number and arrangement of components illustrated in the above block diagrams. It is intended that the appended claims include such changes and modifications.